64-11

435039

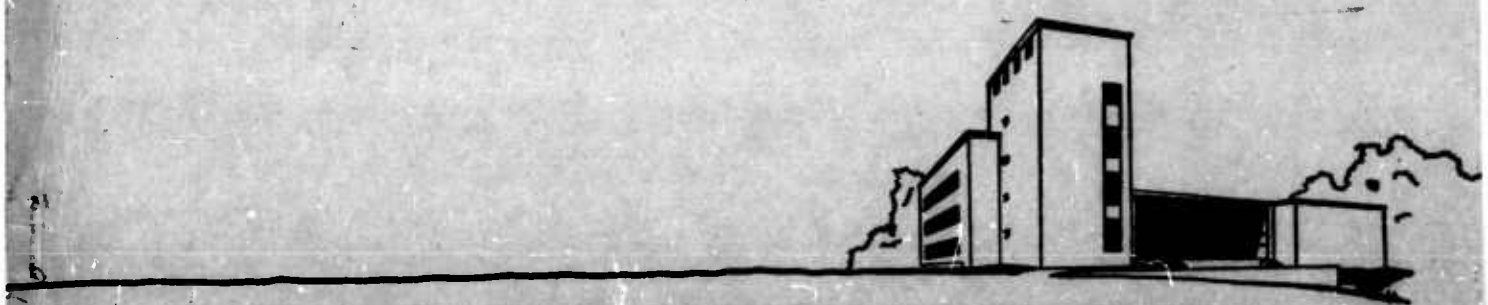
435039

AD No._____

DDC FILE COPY

Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania



GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

378 #3.60

(14) Rept. no. ONR/RM 115

~~O.N.R. Research Memorandum No. 115~~

(6)

ON THE ALGORITHMS OF SZWARC AND FEDEROWICZ FOR LINEAR PROGRAMMING PROBLEMS WITH BOOLEAN VARIABLES*

(10)

by

Robert Connelly

(11) September, 1963

(5) 169 200

Graduate School of Industrial Administration, Pittsburgh, Pa.

Carnegie Institute of Technology

~~Pittsburgh 13, Pennsylvania~~

* This paper was prepared under the direction of Professor G. L. Thompson.

This paper was written as part of the contract, "Planning and Control of Industrial Operations," with the Office of Naval Research, at the Graduate School of Industrial Administration, Carnegie Institute of Technology. Reproduction of this paper in whole or in part is permitted for any purpose of the United States Government. Contract Nonr-760(01), Project NR-047011.

AI

(15) (16)

I. A SUMMARY OF AN ALGORITHM PROPOSED BY A. J. FEDEROWICZ FOR THE FIXED COST LINEAR PROGRAMMING PROBLEM

In his paper, Federowicz proposes an algorithm for solving the "fixed cost linear programming problem" where the problem is the same as in the linear programming problem except that in the objective function there is a fixed cost associated with turning a variable on. It ~~will be~~ is the purpose of this paper to summarize this method and another one solving a closely related problem and to compare the two. ~~It turns out that~~ Federowicz's problem is a special case of the other problem ~~and thus we shall present, later on,~~ so a generalization of this problem is presented.

A. The problem

The first problem considered by Federowicz was:

$$\text{Minimize } \sum_{j=1}^{r} c_j x_j + \sum_{j=1}^{n} c'_j \delta$$

$$\text{Subject to } \sum_{j=1}^{n} a_{ij} x_j \geq b_i \qquad i = 1,\ldots m$$

$$\delta_j = 0 \quad \text{if } x_j = 0 \quad j = 1,\ldots n$$

$$\delta = 1 \quad \text{if } x_j > 0 \quad j = 1,\ldots n$$

$$x_j \geq 0 \qquad j = 1,\ldots n$$

This problem has applications with the problem of scheduling which of a certain number of faculties should be turned on and how much, if a fixed charge is to be paid for turning them on, and other applications (see [1] p. 10).

B. The Method - A General Description

First consider the domain where the variables are defined, i.e., $x_j \geq 0$ $j=1,2,\ldots n$. We shall call a subset of the domain a subset that has

the property that $x_j=0$ or $x_j>0$ $j=1,2,\ldots n$. Thus $\{x|x_1 = 0, x_2 = 0, x_3,\ldots x_n>0\}$ is a subset. Now consider the function $C(x) = \sum c_{-j}x_{-j} + \sum c_j x_j$ defined on each of the subsets. It is linear and convex there and hence we can use the simplex method on each of the 2^n subsets and take the minimum of all of these. This, however, is too lengthy and we must find a method that will solve the problem without necessarily looking at all of the subsets. To this end, consider the union of a certain number of the subsets such that the union is convex. We will call this a group. Now consider just the variables x_j such that $x_j \geq 0$ or $x_j > 0$, i.e., the ones that are definitely turned on and the ones that may or may not be turned on. Now consider $C(x)$ and the constraints where all the other $x_j = 0$, and consider <u>this</u> problem solved by the simplex method, disregarding the effect of the fixed costs. Now if we add just the fixed costs of those x_j definitely turned on we obtain a lower bound on the final minimum, on this group, because the only way we can get a lower cost is to turn off one of the variables $x_j \geq 0$ and not incur its fixed cost, but we never add it, thus it is a lower bound. Or, we can think of the function, $\bar{C}(x) = a_j x_j + f(x)$ where $f(x)$ is just the sum of the fixed charges definitely on, as a lower bounding function on $C(x)$ in the group. Thus the algorithm is as follows: First we consider the whole space $x_j \geq 0$ as a group, then obtain a lower bound and use the actual value obtained as an upper bound on the final minimum. We then remove the subset found by the simplex algorithm where the minimum occurs of the bounding function and partition the rest of the group. We find a lower bound and a feasible solution on each of these and take the one that has the lowest lower bound. We then remove the subset as before and continue on, each time partitioning the group that has the lowest lower bound until we finally have a lower bound

on the remaining groups (the whole set minus the removed subsets) that exceeds the value of one of the removed subsets. Then this removed subset with the lowest minimum is the solution since all the removed subsets have a greater minimum and we have a lower bound that exceeds this minimum for the groups that are left.

C. The Algorithm Used

Let $C[X] = \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} f_j \delta_j$ and X is some convex of the 2^n subspaces, and let $\bar{C}(X) = \sum_{j=1}^{n} c_j x_j + f(x)$ where $f(x)$ is the sum of the fixed costs of the variables definitely turned on. Let S be a rule for determining how to subtract one of the 2^n subsets from some group of them and dividing the remainder into groups.

Let X_Ω be the set of all the 2^n subsets and X_n a particular group of the 2^n subsets.

The algorithm is as follows:

1. Minimize $\bar{C}(X_\Omega)$ by the simplex method, let the subset on which this occurs be X'_1.
2. Set $p = 1$, $X_{q_p} = X$, $r = 0$.
3. Apply S to $X_{q_p} - X'_p$.
4. Let $X_{q_p} = X'_p + X''_{r+1} + X''_{r+2} \cdots X''_{r+n_p}$
5. Set $I = 1,2,\ldots n_p$
6. Minimize $\bar{C}[X''_{r+I}]$ by the Simplex Method, let the subset on which this occurs be X'_{r+I}
7. Set $r = r + n_p$, $p = p + 1$

8. Find $\min_{k=1,\ldots n} \{\min C[x_k']\} = C_T$ min.

9. Find $\min_k \{\min \tilde{C}[x_k'']\} = C_p$, min where k excludes previous C_p min.

10. If C_{p_1} min $\geq C_T$, min then we are done and C_T, min is the answer.

11. Otherwise, let the k from which C_p, min occurred be denoted by k_p. Let $X_{q_p} = X''_{k_p}$, $X'_{k_p} = X'_p$

12. Go to step 3.

It can easily be seen that the algorithm will terminate because there are only a finite number of subsets to subtract, and eventually they will all be looked at if a solution is not found before. This is a specialization of a more general algorithm given by Federowicz, pp. 65-67 in [1]. The following example is the one worked out by Federowicz in [1].

D. Example

An example will serve to illustrate and clarify what has been said. The problem is to minimize $C(x) = 1000x_1 + 1000x_2 + 1000x_3 + 300\delta_1 + 700\delta_2 + 400\delta_3$

where $\delta_j = 0$ if $x_j = 0$

$= 1$ if $x_j > 0$

subject to $x_j \geq 0 \quad j = 1, 2, 3$

$$x_1 + 2/3\, x_2 + x_3 \geq 1$$

$$1/2\, x_1 + x_2 + 2/3\, x_3 \geq 1$$

We may now divide the space into 8 subsets where the cost function and the constraints are linear. Thus one can solve each of these problems in each of the 2^3 or more generally 2^n subsets. They are:

1. $x_1 = x_2 = x_3 = 0$
2. $x_1 \geq 0$, $x_2 = x_3 = 0$
3. $x_2 > 0$, $x_1 = x_3 = 0$
4. x_1, $x_2 \geq 0$, $x_3 = 0$
5. $x_3 \geq 0$, $x_1 = x_2 = 0$
6. x_1, $x_3 \geq 0$, $1/2 = 0$
7. x_2, $x_3 \geq 0$, $x_1 = 0$
8. x_1, x_2, $x_3 \geq 0$

Enumeration of all of these subsets is as follows:

Subset	Minimum Point	Minimum of C(x)
1	No solution	------
2	(2,0,0)	2300
3	(0,3/2,0)	2250
4	(1/2,3/4,0)	2250
5	(0,0,3/2)	1900
6	(0,0,3/2)	2200
7	(0,3/5,3/5)	2300
8	(0,3/5,3/5)	2600

It will be noticed that on subsets 6 and 8 $x_1 = 0$ whereas this is outside the region of definition. This is remedied by letting x_1 be arbitrarily small but positive. Thus we see that we may come as close to the specified cost as we wish. In other words, this cost represents the greatest lower bound of all the costs in that region. It will also be noticed that any such subsets cannot have the solution because by turning off the variable that becomes arbitrarily small we do not incure its fixed cost, and the cost is thus smaller in some other region.

From the table one can see that the minimum is 1900 and it occurs at (0,0,3/2). To find this, however, we had to look at all of the $2^n = 2^3 = 8$ subsets, which is much too laborious. What we would like is a method which looks at far fewer subsets and can tell it is done without necessarily looking at them all.

Let us first denote a group of the 8 subsets by B_n and B_o will represent $\{\vec{x} | x_1, \geq 0, x_2 \geq 0, x_3 \geq 0\}$. $C_{B_n}(x)$ in general will be $= \sum_{j=1}^{n} c_j x_j + f(x)$ where $x \; B_n$ and $f(x)$ is the sum of only those fixed costs definitely turned on. Thus, $C_{B_o}(x) = 1000(x_1 + x_2 + x_3)$. Also it is noted that $C(x) \geq C_{B_o}(x)$ for $x \; B_o$ which is the union of all of the 8 subsets. Therefore the minimum of C_{B_o} may serve as a lower bound on the minimum of $C(x)$. The minimum of C_{B_o} may be found by the simplex method, and we find it is at $(0, 3/5, 3/5)$ and $C_{B_o}(x_{min}) = 1200$ and $C(x_{min_{B_o}}) = 2300$. We now remove set 7 from consideration and divide the remaining subset into three groups, defined as follows:

Set Nos.	Set	Lower Bounding Function
1,2,5,6	$B_1 = \{\vec{x} \; x_1 \geq 0, x_2 \geq 0, x_3 \geq 0\}$	$C_{B_1}(x) = 1000(x_1 + x_3)$
3,4	$B_2 = \{\vec{x} \; x_1 \geq 0, x_2 > 0, x_3 > 0\}$	$C_{B_2}(x) = 1000(x_1 + x_2) + 700$
8	$B_3 = \{\vec{x} \; x_1 > 0, x_2 > 0, x_3 > 0\}$	$C_{B_3}(x) = 1000(x_1 + x_2 + x_3) + 1400$

The rule S for choosing the B_n will be discussed later. It can be seen that it has a great deal to do with the success of the algorithm. Now using the simplex algorithm on these three functions we find:

Bounding Function	Minimum Point	Min $C_{B_n}(x)$	$C(x)$ where $C_{B_n}(x)$ is minimized
C_{B_1}	(0,0,3/2)	1500	1900
C_{B_2}	(1/2,3/4,0)	1950	2250
C_{B_3}	(0,3/5,3/5)	2600	2600

Now we see since the minimum of C_{B_1}, C_{B_2}, C_{B_3} is 1500 we have now a new lower bound on the minimum, $C(x)$ on the subsets 1,2,...6,8. Also we have a new value for $C(x)$ of 1900, which was better than before. Now since this minimum occurred on subset 5 we remove it from B_1 and look at the rest, i.e., 1,2, and 6. We divide them as follows:

Set Nos.	Set	Lower Bounding Function
1,2	$B_4 = \{\vec{x} \mid x_1 \geq 0, x_2 = 0, x_3 = 0\}$	$C_{B_4}(x) = 1000x_1$
6	$B_5 = \{\vec{x} \mid x_1 > 0, x_2 = 0, x_3 > 0\}$	$C_{B_5}(x) = 1000(x_1 + x_3) + 700$

And solving each of these we find:

Bounding Function	Min. Pt.	Min. of C_{B_i}	Corresponding Min $C(x)$
$C_{B_4}(x)$	(2,0,0)	2000	2300
$C_{B_5}(x)$	(0,0,3/2)	2200	2200

Now we find that the minimum of C_{B_2}, C_{B_3}, C_{B_4}, C_{B_5} is 1950, and therefore this may serve as a lower bound on the minimum of $C(x)$ on 1,2,3,4,6,8. The best $C(x)$ found, however, was 1900 on subset 5. Therefore, we may conclude that (0,0,3/2), the minimum point found on 5, is the minimum of $C(x)$ since $C(x) \geq 1950$ on 1,2,3,4,6,8 and $C(x) \geq 2300$ on subset 7.

It should be observed that on each of the 2^n subsets $C(x)$ is linear and convex and thus the problem may be solved completely on any one of them. However, on a group of subsets such as the B_n one can only bound the $C(x)$ with a linear convex function, which allows us to use the simplex algorithm.

Also we note that the sequence 1200,1500,1950 may be identified with C_P min and the sequence 2300,1900,1900 as C_T min in section C. Also, the $C_{B_i}(x)$ with $\bar{C}[X_k^n]$ and $\Omega = B_o$.

E. Reduction of Computations

In the preceding example it will be noted that six times a function had to be minimized by the simplex method. This does not seem like much of an improvement over the original eight times. However, we need not start from scratch each time. When a minimum is found, to go to a different region we need only eliminate the appropriate basic element by pivoting on its row and then removing its column. An example will illustrate and clarify. The original tableau for the problem is:

Faculty	1	2	3	4	5
Fixed Costs	300	700	400	0	0
Unit Costs	1000	1000	1000	0	0
Requirements			Matrix		
1	1	2/3	1	-1	0
1	1/2	1	2/3	0	-1

Solving this we find:

	Faculty	1	4	5
	Fixed Costs	300	0	00
	Unit Costs	1000	0	0
Faculty	Fixed-Cost	Unit Costs	Amount	Matrix
3	400	1000	3/5	6/5 -9/5 6/5
2	700	1000	3/5	-3/10 6/5 -9/5
		Marginal Costs		-100 -600 -600

If we wish now to find min $[C_{B_1}(x)]$ we must eliminate faculty 2 from the basis and minimize the result. Only 4 can be used to eliminate 2, thus we must pivot on 6/5 and forget about that column. The result is:

	Faculty	1	5
	Fixed Costs	300	0
	Unit Costs	1000	0

Faculty	Fixed Costs	Unit Costs	Amount	Matrix	
3	400	1000	3/2	3/4	-3/2
4	0	0	1/2	-1/4	-3/2
		Marginal Costs		-250	-1500

F. The Set Subtractions Problem

Previously it was stated that there was a rule (S) for determining how to break up the groups of subsets into smaller groups and just one of the subsets. The rule described by Federowicz will now be presented (see pp. 68-71 in [1]). First, however, we will need some new notation. We will denote a group of subsets by a series of B_i's and $\bar{B}_i$'s "multiplied" together. The presence of B_i will indicate x_i is definitely on. The presence of $\bar{B}_i$ will indicate $x_i = 0$. If neither is printed it will mean $x_i \geq 0$. Thus, $B_1 B_2 \bar{B}_3$ is the subset 4, $x_1, x_2 > 0, x_3 = 0$. $B_1 \bar{B}_3$ is the union of subsets 4 and 2, i.e., $x_1 > 0$, $x_2 \geq 0$, $x_3 = 0$, etc. Thus we can see that any two different representations by this notation represents two different convex groups of subsets, and conversely.

Now it can be observed with this notation that there are a number of ways of dividing any group. For instance:

$$\cap - \bar{B}_1 B_2 B_3 = B_1 + \bar{B}_2 \bar{B}_1 + \bar{B}_3 B_2 \bar{B}_1$$
$$= B_1 + \bar{B}_3 \bar{B}_1 + \bar{B}_2 B_3 \bar{B}_1$$
$$= \bar{B}_2 + B_1 B_2 + \bar{B}_3 \bar{B}_1 B_2$$
$$= \bar{B}_2 + \bar{B}_3 B_2 + B_1 B_2 B_3$$
$$= \bar{B}_3 + \bar{B}_2 B_3 + B_1 B_2 B_3$$
$$= \bar{B}_3 + B_1 B_3 + \bar{B}_2 \bar{B}_1 B_3 ,$$

and it will be seen that each division corresponds to a particular ordering of the variables. For consider

$$\cap - B_{n_1} B_{n_2} \cdots B_{n_m} = \bar{B}_{n_1} + B_{n_1} \bar{B}_{n_2} + B_{n_1} B_{n_2} \bar{B}_{n_3} \cdots + B_{n_1} B_{n_2} \cdots \bar{B}_{n_m}$$

where $B_{n_i} = B_j$ or $\bar{B}_j$ for some j and $\bar{B}_{n_i} = \bar{\bar{B}}_j = B_j$ if $B_{n_i} = \bar{B}_j$, and similarly for the other set subtractions.

The rule for determining the ordering of B_j is as follows: Let $\sum - \bigcap_{j=1}^{n} B_j^*$ be the set subtraction under consideration, and C and $\bar{C}$ be the same as before and f_j the fixed costs. Then we divide the n variables into three classes.

Class 1: Set of variables for which $B_j^* = \bar{B}_j$ and $\bar{C}(X_{\sum}) + f_j \geq C(X_{\sum})$

Class 2: Set of variables for which $B_j^* = B_j$

Class 3: Set of variables for which $B_j^* = \bar{B}_j$ and $\bar{C}(X_{\sum}) + f_j \leq C(X_{\sum})$

Class 1 consists of those variables which were off in the linear programming solution and whose fixed cost plus the lower bound exceeds the actual cost obtained. Since by turning the variable x_j on we incur at least f_j we know that it must not be turned on. Therefore, Class 1 consists of those variables that were turned off and should remain off.

Class 2 consists of those variables which were turned on, and they are placed in the inverse order of our estimate of the minimum cost incurred in turning each one off.

Class 3 variables are last, and it is difficult to see why any ordering is better than any other since by turning the variable on we have a lower bound lower than actually obtained.

G. Incomplete Solutions

It can easily be seen that if the algorithm is stopped before it is finished, the current minimum on the removed subsets and the minimum of the lower bounds on the other subsets serve as upper and lower bounds respectively on the solution.

In the computation of most of the problems worked out by the computer, Federowicz found that the solution was found relatively quickly and what took most of the time was obtaining a lower bound greater than the current minimum found, i.e., verifying that the minimum has been found. For example (p. 74 in [1]) one problem took 31 revisions to find the minimum and 85 further revisions to verify it. Another problem took 4 revisions to find the minimum and 41 to verify it. Thus one is tempted, for larger problems, to quit after the same minimum has been found a certain predetermined number of times, and use that minimum as a solution.

II. GENERALIZATION OF FEDEROWICZ'S METHOD TO SZWARC'S PROBLEM

Federowicz considers a generalization of his algorithm (see p. 65 in [1]) and to solve Szwarc's problem we must apply the generalization rather than the actual algorithm previously considered. Szwarc's problem is as follows:

$$\text{Maximize} \quad z = \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} \leq b_i \quad \text{and} \quad x_j = 0 \text{ or } 1 \text{ for } j = 1, 2 \ldots p$$

$$i = 1, 2 \ldots m \,, \; 0 \leq x_j \leq d_j \text{ for } j = p+1, \ldots n$$

Now one can see that the algorithm will be somewhat different from before and is as follows: First we consider the 2^p subsets as before and the problem defined on each of them. It is easy to see that each one could be solved by the simplex method and we could then take the best solution. However this is too lengthy. As before we instead consider a group of these subsets where some are definitely 0, some definitely 1, and others either 0 or 1, and we shall obtain an upper estimate on the final maximum. Now suppose we have such a group of subsets. If $x_j (j \leq p)$ is definitely 0 or definitely 1 it will remain so in the objective function and the constraints. If $x_j \geq 0 (j \leq p)$ x_j will be considered to be 1 in $z(x)$, the objective function, if $c_j \geq 0$ and x_j will be considered to be 0 if $c_j \leq 0$. The lower estimate, similar to before, shall be the value of the basis that is feasible and satisfies the Boolean constraints. For the upper estimate we now need only worry about the constraints. For each constraint inequality we will derive a new one which will amount only to changing the b_i if we only work with

$x_{p+1} \cdots x_n$ in the simplex solution. That is $B_i = b_i - \sum_{j=1}^{p} a'_{ij}$ where

$$a'_{ij} = \begin{cases} a_{ij} & \text{if } x_j \text{ is 1 definately, or if } a_{ij} \leq 0 \text{ and } x_j \text{ is in doubt} \\ 0 & \text{if } x_j \text{ is 0 definitely, or if } a_{ij} \geq 0 \text{ and } x_j \text{ is in doubt} \end{cases}$$

Thus with B_i instead of b_i and considering only $x_{p+1}, \cdots x_n$ we can now solve the linear programming problem using the simplex method, and this serves as an upper bound since any value of the x_j's in doubt that satisfy the original constraints, satisfy the new constraints.

A. Reduction of Computation

As before we need not make a new tableau and solve each one from scratch in the simplex algorithm. If we record each $a_{ij} (j \leq p)$ and evaluate it after the first minimization, i.e. of the lower bound. We may then pivot to get rid of those values of x_j in the basis which become negative when the appropriate a_{ij} change, and maximize the resulting tableau. Also in changing from group to group we need only pivot to get rid of the negative B_i, and then maximize the resulting tableau.

B. Set Subtraction Problem

Now we must determine which subset to extract from the group that is to be divided and how to break it up. Before, the simplex solution gave the subset to be extracted but in our case this need not be the case, and in fact would not be desirable since we know beforehand on which subset our feasible maximum comes from (where all the x_j in doubt = 1). Thus the only thing that would distinguish our algorithm from enumeration (if we were to always to extract that subset) would be the ordering used to determine how to break up the group and which group has the greatest upper bound. Thus to gain greater freedom some better

way of determining which set to extract would be in order. One method might be to turn off those variables in the matrix that cause the total increase in the B_1 of the matrix minus its fixed charge in z if it is negative and plus if it is positive to be the greatest. Those with a positive increase would be turned off and those with a negative increase kept on.

As for subtracting the set we have now just to investigate which ordering should be used to determine how to break up the group of subsets in question. In contrast to the example we should use more information than just the fixed cost in the objective function. We may put those variables first which are on and which their fixed cost in the objective function subtracted from the upper bound minus some estimate of the decrease in the objective function due to the other variables is smallest. Thus using this total estimate we may order all the variables in this problem and determine the resulting groups of subsets as before.

C. Conclusion

One will notice that this generalization is somewhat different from the original problem, because of the added Boolean constraints and because in the solution in finding the estimates, just the domain of definition varies, not the objective function. And when extracting a subset from a group we still must find the maximum on the subset extracted. It is not automatically provided.

D. Original Problem As A Special Case

It should be noted that the first problem may be stated as a special case of Szwarc's problem (with min replaced by max) as follows:

$$\text{Maximize} \quad \sum_{j=0}^{n} c_j \delta_j + \sum_{j=0}^{n} c'_j x_j$$

$$\text{subject to} \quad -d_j \delta_j + x_j \leq 0 \qquad j = 1, 2 \ldots n$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$$

where $\delta_j, x_j \geq 0$ and $\delta_j = 0$ or 1 and the d_j are big enough.

III. THE GENERALIZATION OF FEDEROWICZ'S METHOD APPLIED TO AN EXAMPLE SOLVED BY FEDEROWICZ'S ORIGINAL METHOD.

The problem is to minimize $z = 300x_1 + 700x_2 + 400x_3 + 1000(x_4+x_5+x_6)$

subject to

$x_4 + 2/3\ x_5 + x_6 \geq 1$ $\qquad$ $2x_1 - x_4 \geq 0$

$1/2\ x_4 + x_5 + 2/3\ x_6 \geq 1$ $\qquad$ $3/2\ x_2 - x_5 \geq 0$

$3/2\ x_3 - x_6 \geq 0$

The dual tableau will look like:

		y_1	y_2	y_3	y_4	y_5
x_1	300	0	0	2	0	0
x_2	700	0	0	0	3/2	0
x_3	400	0	0	0	0	3/2
x_4	1000	1	1/2	-1	0	0
x_5	1000	2/3	1	0	-1	0
x_6	1000	1	2/3	0	0	-1
x_6	1000	-1	-1	2	3/2	3/2

And after pivoting

		x_6	x_5	y_3	y_4	y_5
x_1	300	0	0	2	0	0
x_2	200	0	0	0	3/2	0
x_3	400	0	0	0	0	3/2
x_4	100	-6/5	3/10	-1	-3/10	6/5
y_2	600	-6/5	9/10	0	-9/5	6/5
y_1	600	9/5	-6/5	1	6/5	-9/5
	1200	3/5	3/5	2	9/10	-9/10

Thus we now see that we have a lower bound of 1200 on Ω just as in the original problem. Now, because of the nature of the matrix we may subtract $\bar{B}_1$ B_2 B_3 from Ω. We do this because we see that y_3 would be 0, if x_1 were 0, and this is not so with any of the others. Thus, on $\bar{B}_1$ B_2 B_3 2300 is the minimum and this would be the first element of the decreasing sequence. Now suppose we wish to find the minimum on $\bar{B}_2$. To do this, we first change the last row only in y_4 column. By subtracting 3/2 we see that the 9/10 changes to -3/5, thus we pivot on this column to obtain a new lower bound of 1500 as before, and then we continue on to the other groups. Thus one can see that with the correct set subtraction we can obtain the same algorithms as before, only the matrix is different and the rules for pivoting from group to group are different. However, the resulting two sequences are identical, if the set subtractions are the same.

IV. SZWARC'S METHOD

In his paper [2] Szwarc presents a completely different method from Federowicz's in solving the mixed integer linear programming problem. Federowicz's method, roughly speaking, is to start with a feasible maximum and continually get better ones until the estimates indicate that the right one is found. Thus he works his way up through feasible maximi. Szwarc's method is just the opposite. He works his way down through infeasible maximumi until a feasible solution (one satisfying the Boolean constraints) is found and thus the computation is ended.

A. The Problem

A slightly more general formulation of the problem than previously stated is:

$$\text{Maximize} \quad \sum_{j=1}^{n} c_j x_j = z$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \quad x = 1, 2 \ldots m \tag{1}$$

$$0 \leq x_j \leq 1, \; j = 1, 2, \ldots p \leq n, \; 0 \leq x_j \leq d_j \; (d_j \geq 0),$$

$$j = p + 1, \ldots n. \tag{2}$$

$$x_j = \begin{cases} 0 \\ \text{or} \\ 1 \end{cases} \qquad j = 1, \ldots p. \tag{3}$$

We shall call this problem I. This is the problem we eventually wish to solve.

The first part of condition (2) may seem unneeded considering condition (3), but we shall form another problem, called problem II, where we only have conditions (1) and (2). That is for problem II we maximize the same function as problem I but with just the conditions (1) and (2).

Now suppose problem II is solved and it has a maximum of z_o. We shall now formulate another problem, problem III, which is to:

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j = z$$

$$\left.\begin{array}{ll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i = 1, \ldots m \\ & \sum_{j=1}^{n} c_j x_j = z_o - \delta - t \end{array}\right\} \quad (1')$$

$$0 \leq x_j \leq 1, \; j = 1, \ldots p \leq n, \; 0 \leq x_j \leq d_j \quad j = p+1, \ldots, n \qquad (2)$$

where t is a nonnegative parameter and δ is sufficiently small. It will be noticed that the maximization is not needed since the last condition in (1') insures that it will be at a maximum. Now one can see that this problem is a parametric linear programming problem (see pp 220-231 in [4]) and therefore we can find a sequence

$$t_1, t_2, \ldots, t_k \;, \quad 0 \leq t_1 \leq t_2 \geq \ldots \leq t_k$$

and the corresponding sequence

$$x_1(t), x_2(t) \ldots, x_{10}(t)$$

where $x_s(t)$ is a solution to problem III for

$t_{s-1} \leq t \leq t_s$ $(t_o = 0)$ and the bases of two consecutive solutions differ by just one variable. The way the sequence is developed is as follows:

First we solve the problem with t = o (or arbitrarily close to it and positive). Then we can see that each variable in the basis is a linear function of t. Then we increase t until one of the variables becomes negative. Then we pivot on the row of that variable (see pp 220-231 in [4]) such that the basis remains optimum. Thus we obtain a new basis for a new largee value of t. We then continue in continually increasing the value of t and obtaining new bases such that each successive one differs by one element, and thus obtain the sequence previously described.

If there are several feasible solutions for $x_s(t)$ we must consider them all.

We also know that $x_s(t)$ is a linear function of t in the interval $[t_{s-1}, t_s]$.

Now it can be proved (see Scwarcz Appendix in [2]).

1. The minimum value of t such that $x_s(t)$ satisfies the Boolean constraints is the solution to problem I. This is easily seen by looking at consition (1)'.

2. The solution to problem I occurrs at a corner point of problem III. That is either $x_s(t_{s-1})$ or $x_s(t_s)$ satisfies the Boolean constraints if some t in $[t_{s-1}, t_s]$ allows $x_s(t)$ to satisfy the Boolean constraints, and if $x_s(t_s)$ satisfies the Boolean constraints t_s is the unique value of t or $x_s(t_{s-1})$ also satisfies the Boolean constraints.

B. The Algorithm

We can now see more or less how to solve the problem with the help of these last two statements. The following, however, is an algorithm proposed by F. M. Tonge in his paper [3].

1. Find the solution to problem II, (in general there will be only 1 and it will not satisfy the Boolean constraints. If it does, however, we are finished) and consider problem III.

2. Establish a list L which is to be a list of extreme points, in a set of basis vectors $x_i(t)$, and <u>always keep it increasing order of t max</u>.

3. Find each optimal basic solution to problem, and find all first intervals, in the $x_i(t)$ sequences, and find the t max for that interval and enter it in the appropriate spot in the list L.

Also check to see if this entry in the list satisfies the Boolean constraints and if it does delete all entries in the list that have a greater t max.

That is, if i' is the row of the constraint becoming negative, enter in the list L the basis formed by replacing i' by j' with t max given by

$$t\ \max = \min_{i = d_i - \frac{a_{ij'}}{a_{i'j'}} \quad d_i < 0} \left\{ \frac{b_i - a_{ij'} b_{i'}}{-(d_i - \frac{a_{ij'}}{a_{i'j'}} d_{i'})} \right\}$$

where d_i is the coefficient of t in equation i, and $a_{i'j'} < 0$.

4. Consider the first entry in the list L. If it has satisfied the Boolean constraints (it will have been checked before) we are done. If not move to step 5.

5. Remove the first entry from the list, then for that entry find all the possible next intervals and enter their basis in the list as in step 3, checking for Booleaness. Then return to step 4.

If no further intervals can be found, no solution to problem I exists.

It should be noted that due to the nature of constraint (1') there will in general be a number of first, second, etc. intervals. In fact, this is the main disadvantage of the algorithm .

It was pointed out by Tonge that in the list L only changes in the bases need be considered in each entry, (see page 3 in [3]).

V. SZWARC'S METHOD APPLIED TO AN EXAMPLE SOLVED BY FEDEROWICZ'S METHOD

Szwarc's algorithm is primarily concerned with a maximization problem. However, it is easily seen how to apply it to the minimization problem, in particular the example considered by Federowicz. The rest of this paper, however, will be concerned with the maximization problem.

The problem again is to:

Minimize $C_{(x)} = 1000 (x_1 + x_2 + x_3) + 300 \delta_1 + 700 \delta_2 + 1400 \delta_3$

where $\delta_j = 0$ if $x_j = 0$

$= 1$ if $x_j > 0$

subject to $x_j \geq 0$

$$x_1 + 2/3\, x_2 + x_3 \geq 1$$

$$1/2\, x_1 + x_2 + 2/3\, x_3 \geq 1$$

Or in Szwarc's form:

Minimize $C_{(x)} = 300x_1 + 700x_2 + 400x_3 + 1000 (x_4 + y_5 + x_6)$

subject to $x_j \geq 0$

$-x_1 \geq -1$	$2x_1 - x_4 \geq 0$	$x_4 + 2/3\, x_5 + x_6 \geq 1$
$-x_2 \geq -1$	$3/2\, x_2 - x_5 \geq 0$	$1/2\, x_4 + x_5 + 2/3\, x_6 \geq 1$
$-x_3 \geq -1$	$3/2\, y_3 - x_6 \geq 0$	
(1)	(2)	(3)

where x_1, x_2, x_3 are 0 or 1. This is problem I. Problem II is the same, only without the last constraint. Problem III is the same as problem II, except we have the paremetric constraint $300x_1 + 700x_2 + 400x_3 + 1000 (x_4 + x_5 + x_6) \geq z_o + t$ where z_o is the minimum found by problem II.

The constraints (1) are those that insure that the Boolean variables do not exceed 1. It is hoped that they will turn out to be 1 or 0 in problem III.

The constraints (2) are those that insure that the Boolean variables will be turned on with their correct counterparts. x_1, x_2, x_3 correspond to g_1, g_2, g_3 and x_3, x_4, x_6 to y_1, y_2, x_3 in the previous problem. The coefficients of x_1, x_2, x_3 in (2) are obtained from the constraints in the previous problem.

The constraints (3) are identical to before. The initial tableau for problem II will look like:

Facility		1	2	3	4	5	6
				Matrix			
Restraints							
(1)	- 1	-1	0	0	0	0	0
	- 1	0	-1	0	0	0	0
	- 1	0	0	-1	0	0	0
(2)	0	2	0	0	-1	0	0
	0	0	3/2	0	0	-1	0
	0	0	0	3/2	0	0	-1
(3)	1	0	0	0	1	2/3	1
	1	0	0	0	1/2	1	2/3
costs		+300	+700	+400	+1000	+1000	+1000

Rewriting this in the dual form and maximizing, we have:

x_2	520	-6/5	9/5	6/5	-4/5	0	-6/5	4/5	0
x_1	$\frac{3320}{3}$	9/5	-6/5	-4/5	22/15	0	4/5	-22/15	0
y_4	$\frac{2720}{3}$	3/8	-6/5	-4/5	7/15	-1	4/5	-7/15	0
x_3	$\frac{800}{3}$	0	0	2/3	0	0	-2/3	0	0
x_4	$\frac{1400}{3}$	0	0	0	2/3	0	0	-2/3	0
y_1	300	0	0	0	0	2	0	0	-1
1640		3/5	3/5	2/5	2/5	0	3/5	3/5	1

Thus the solution to problem II is (0, 2/5, 2/5, 0, 3/5, 3/5) and the minimum is 1640 which is less than 1900, the minimum with the added constraint of x_1, x_2, $y_3 = \begin{cases} 0 \\ 1 \end{cases}$ (as known from Federowicz's problem). Now we must add the new constraint $C(X) = z_o + t$. With this added column, we see that we repeat the same final column as before, only -t will be in the last row (first element). We must now pivot on this column with each of the entries. After doing this we see that all the entries in the last row are positive for t is sufficiently small in all cases, except one where we pivot on the y_4 row. Since the new entries in the c_j column are equal to $c_j + \frac{a'_{ij}}{a'_{ij'}}$ where a_{ij} is the pivot element of the newly added column. Thus the t_{max} for this pivot will be $t_{max} = \min \frac{-c_j a_{i'j'}}{a_{i'j}}$ where $a_{i'j} < 0$.

Thus we have:

basic change from y_5, y_6, y_3, y_2, y_5, x_6, x_7, y_8, R

No.					
1	x_2 - > R	---	260	---	x_6 and y_5 and y_2
2	x_1 - > R	---	$452\frac{8}{11}$	---	x_5
3	x_3 - > R	---	600	---	x_6
44	x_4 - > R	---	420	---	x_7
5	y_1 - > R	---	300	---	y_8

Now by pivoting on the x_5 column we do not change the objective (i.e., it is another min point) function, and then pivoting in the R column and x_5 we now may add to the table.

6	y_1 - > x_5, $/4$ - > R		$528\frac{}{3}$	---	y_6 and y_3

Thus L will be, in order of the number of basis, 1, 5, 4, 2, 3. Thus the first one looked at is 1. Pivoting we have with the new column:

(see following page).

Now we can see that at $t = t_{max} = 260$ we have 0 or 1 as the first three elements, and thus (0, 0, 1, 0, 0, 3/2) is the minimizing point. Since there were no elements in the list with a t_{max} less than 260, this is the solution to the problem, and the minimum is 1900, the same as obtained by Federowicz.

		x_2	y_5	y_6	y_3	y_2	x_5	x_6	x_7	x_8
R	1	$\frac{1}{520}$	$-\frac{3}{1300}$	$\frac{9}{2600}$	$\frac{3}{1300}$	$\frac{-1}{650}$	0	$-\frac{3}{1300}$	$\frac{1}{650}$	0
							0			0
x_1	0	$-\frac{332}{156}$								
y_4	0	$\frac{272}{156}$					-1			0
x_3	0	$\frac{-80}{156}$					0			0
x_4	0	$-\frac{140}{156}$					0			0
y_1	0	$\frac{-30}{156}$					2			-1
	$1640+t$	$+\frac{t}{520}$	$\frac{3}{5}-\frac{3}{1300}t$	$\frac{3}{5}+\frac{9}{5}\frac{1}{520}t$	$\frac{2}{5}+\frac{6}{5}\frac{1}{520}t$	$\frac{2}{5}-\frac{4}{5}\frac{1}{520}t$	0	$\frac{3}{5}-\frac{6}{5}\frac{1}{520}t$	$\frac{3}{5}+\frac{4}{5}\frac{1}{520}t$	1

VI. A COMPARISON OF FEDEROWICZ'S METHOD AND SZWARC'S METHOD FOR SOLVING THE MIXED INTEGER LINEAR PROGRAMMING PROBLEM WHERE THE INTEGERS ARE ZERO OR ONE

Now, having seen both methods and how they have been applied to one specific example, we can say something about the relative merits of the two methods. First, we shall talk about the problem proposed and solved by Szwarc, and compare it to the same problem solved by Federowicz's generalized algorithm . As has been said before, the attack on the problem by both algorithms are roughly the same in that they both use the simplex method as a sub-algorithm and pivot from corner point to corner point in searching for the final maximum. The difference, however, is that Szwarc defines a different and larger problem (problem II) which "contains" the original problem if further restrictions are made (namely that the first p of the variable be 0 or 1), and then searches down through this new convex set of feasible solutions until a solution to the original problem (problem I) is found. This is the effect of the parametric t and the added constraint in problem III. Federowicz's algorithm , however, does quite the opposite. He does not change the original problem and always pivots and remains in basic feasible solutions to the original problem (problem I in Szwarc's algorithm). He starts at the origin, let us say, and then continually increases the best maximum yet found. He continues this way until the estimates on the rest of the set (not necessarily convex) indicate that the rest of the set is less than the current maximum. Thus we can see that Federowicz's algorithm works with a smaller set which is not convex and finds his way up to the maximum, whereas Szwarc works with a larger set, a convex set, and he works his way down to find the solution.

From this we might suspect that Federowicz's method would almost always be quicker (in computational time) to find a solution than Szwarc's method.

However, there are two reasons why this is not necessarily so. First, although the total number of extreme points is increased, this does not guarantee that there will be a larger number between the solution to problems I and the solution to Problem II than extreme points considered by Federowicz's method. Second, not only must the extreme points of the current maximi in Federowicz's method be considered in computational time, but also we must consider computation time expended in applying the simplex algorithm on the groups of subsets used to estimate the maximum on the subsets not actually substracted. Nevertheless, we can say something about the relative computational merits of the two algorithms.

First we should look at the relative computational time of Federowicz's generalized algorithm and his first more special one. On the same problem the generalized algorithm will be slower because the matrix is n + m by 2n if there are n variables and m constraints in the original problem. Thus we see that the tableau we are to work with is increased by $2n^2 + mn$, which could slow the algorithm down considerably and cause even more problems with storage space. This also shows some of the difficulties involved as the number of variables increases. Not only do the number of pivots increase, but also the time it takes to make each pivot.

Now we can use Federowicz's first method as a standard to increase in some sense the other algorithms. We have seen that the generalized algorithm for the same number of on-off variables and the original problem takes the same number of pivots as the standard, but takes a longer time to make them. In the application of Szwarc's algorithm to the standard we see that the matrix is the same size as the generalized algorithm and the question is how operations on pivots are to be made to find the solution. If the example previously exhibited is any indication, we see that this algorithm

is much shorter than either of the other two. However, it is easy to see that there exists many examples where it takes much longer to find the solutions and we must go much deeper into the list L.

Now let us see what happens when we vary the number of Boolean variables relative to the other variables in Szwarc's problem. Let us say that the number of Boolean variables is p and the number of non-Boolean variables is n and there are m constraints. Now let us count the number of extreme points that will be in the total convex set considered by Szwarc's problem II, and see how many fewer there will be in Federowicz's set which is smaller but not convex.

First, let us count the total number of extreme points considered in Federowicz's treatment of the problem. It is easy to see that there are 2^p subsets to consider, and for each of these subsets $\binom{m+n}{m} = \frac{(m+n)!}{n!\,m!}$ = the number of combinations of picking $m+n$ things m at a time] since there are m slack variables, n regular variables and only m variables in the basis. Thus in all there are $2^p\binom{m+n}{m}$ extreme points in the set. Next let us consider the number of extreme points in Szwarc's problem II. Due to the nature of the first p constraints both the variable and its slack variable cannot both be out of the basis, if the variable (or its slack variable) is one of the first p. Thus we have three possibilities for each of these first p variables. Either both it and its slack variable are in the basis, or just the regular variable or just the slack variable. (If x_i is one of the first p variables, $1 - x_i = x_i'$ is its slack variable.) If there are k variables that have both the regular variable and its slack variable in the basis, there are $\binom{p}{k}$ combinations of the variables such that this can happen, 2^{p-k} different

arrangements of the other $p-k$ Boolean variables, and $\binom{m+n}{m-k}$ arrangements of the other non-Boolean variables. Thus, if $p \leq m$, we have

$\sum_{k=0}^{p} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k}$ extreme points and if $p \geq m$, we have

$\sum_{k=0}^{m} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k}$ extreme points or more generally

$\sum_{k=0}^{\min(p,m)} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k}$ extreme points. And thus we see that

$\sum_{k=1}^{\min(p,m)} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k}$ is the number of additional extreme points created by Szwarc's problem II. Let us now obtain a lower bound for this number. First let us suppose that p is small <u>relative</u> to m and n ($p \leq m$ at least). Then $\sum_{k=1}^{p} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k} \geq \min\left[\binom{m+n}{m}, \binom{m+n}{m-p}\right] \sum_{k=1}^{p} \binom{p}{k} 2^{p-k} =$ $\min\left[\binom{m+n}{m}, \binom{m+n}{m-p}\right] (3^p - 2^p)$, $\{\binom{m+n}{m} \leq \binom{m+n}{m-p}$ if and only if $m \geq n + p\}$ or $\binom{m+n}{\max[m,\, n+p]} (3^p - 2^p)$. Thus if m is not too much larger than $n + p$ we see that for large p (but still small relative to m) the new extreme points of Szwarc's problem far exceeds the <u>total</u> number of extreme points in Federowicz's problem, due to the weight of the 3^p. In fact, if $m \geq n + p$, $\binom{m+n}{m} (3^p - 2^p) > 2^p \binom{m+n}{m}$ for all $p > 1$ and the inequality becomes stronger as p increases. Now the problem is to estimate the number of these new extreme points which lie above the set considered by Federowicz. This number will certainly be far less than the total number

of pivots that Szwarc's algorithm finally makes since each of these points must be visited at least once, and probably more often since the algorithm will have to retrace its steps to get to the next lowest vertex. Now, if this number is just as great as the original set considered by Federowicz, i.e., $2^p \binom{m+n}{m}$, by proportion to the total number of points in the set, i.e., $c\,(3^p - 2^p)\binom{m+n}{m}$ [for some constant c] we see that for sufficiently large p Federowicz's method will be much superior, since at the very most it considers $2^p \binom{m+n}{m}$ extreme points and usually not even that many.

Now, let us suppose that p is large relative to m or n. It is difficult to see just what will happen always. However, with Szwarc's method we can see that the additional extreme points do not grow as 3^p as before since in the sum there are only m terms. In fact, the number of additional extreme points is $\sum_{k=1}^{m} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k}$. Whereas in Federowicz's they are still $2^p \binom{m+n}{n}$, which is now getting quite large. In fact, let us say that there is just one original constraint, ie., m = 1, and that $p < 2n + 2$, but still large. Then:

$$\sum_{k=1}^{m} \binom{p}{k}\binom{m+n}{m-k} 2^{p-k} = p\,2^p\,1/2 < (n+1)\,2^p = 2^p \binom{m+n}{n} .$$

Now we see that since $\binom{m+n}{n} = n + 1$ is a relatively small number that the "help" Federowicz's method gets from the simplex algorithm will be relatively small and that the rules for determining how the sets are broken up will tell how fast the algorithm works. If the example is any indication we see that at least somewhere on the order of 2^p pivots will have to be made. In Szwarc's case we see that no more than $p\,2^p\,1/2$ different extreme points will be visited, and there will probably be much less. Thus if too many are not revisited we see that Szwarc's method should be superior.

This brings up an important point. Much depends on just how "good" Szwarc's method is. We have made an estimate of the number of different extreme points to be visited, but obviously much depends on how good this estimate is and how often we revisit an old vertex. If it turns out that many vertises are often revisited, we may be able to remedy the situation by changing the algorithm. It would be to remember instead the tableau at any vertex it might have occasion to return to, and then jump directly from vertex to vertex without having to pivot over and over. Each time a pivot is made it would be to a new vertex. Or the algorithm could be changed differently. It would be a mixture of both ways. That is, only a few "key" vertices would have their tableaus remembered. If we are in one end of the set and we wish to go to the other, we first jump to the tableau of the nearest "key" vertex (nearest in the sense of the number of pivots) and then pivot to the desired vertex. We now see that the revised algorithm will more nearly resemble Federowicz's algorithm due to the storage of the tableaus, and in my opinion present a fairer comparison of the two algorithms.

Now one more interesting special case is when $n = 0$, that is, all the variables are Boolean. In Federowicz's algorithm the number of extreme points is just 2^p. However, there is no application of the simplex algorithm and the problem is just to find a feasible solution. Once again the algorithm's speed depends only on the choice of the rules for set subtraction. In Szwarc's algorithm, however, the simplex algorithm is applied and we may hope for a shorter solution. The one drawback is that the set of extra points is much larger than 2^p. To see this suppose $m \leq p$, then: $\sum_{k=1}^{m} \binom{p}{k}\binom{m}{m-k} 2^{p-k} > p \sum_{k=1}^{m} \binom{m}{k} 2^{p-k} = 2^p\,(p\,2^{-m} - 1) > 2^p$ if m is small enough, i.e., $m \leq \frac{\log p}{\log 2} - 1$. However, we would suspect that if Szwarc's method works at all, it should work in this case, and should compare favorably with Federowicz's method.

Now, if it turns out that both algorithms do fairly well with the same problem, and they both take about the same amount of time, we might be able to obtain a faster algorithm by mixing the two, and use each one to help the other. The way it would work is as follows. At each step or pivot we would make one pivot in the Szwarc algorithm, and then one pivot in the Federowicz algorithm. If in Szwarc's method we come across a possible solution to problem I (but has not pivoted to it yet), that is it has a maximum better than any of the maximi already extracted, and in the group that is to be next broken up we may extract this set instead of those determined by the normal rules. Also, if in Szwarc's method there are members of the list L that have a maximum less than the current value of the best maximum in Federowicz's method, we can eliminate it from the list L, since it will never be pivoted to anyway. Also, if we decide to stop before the final solution is found, the current value of $z_o - t_{max}$ in Szwarc's method or the greatest upper bound in Federowicz's method will serve as an upper bound, and the best point found by Szwarc's method that is feasible in problem I or the current value of Federowicz's method will serve as a lower bound on the final solution. These can be used to estimate the solution if it is computationally infeasible to go on, and it can be seen that the appropriate bounds also apply to each separate algorithm as well as the composite one just described. Also, instead of applying them both at once one might be inclined to try first one then the other if the first proves infeasible computationally.

Lastly, it is easy to see that both methods can be used to find secondary maximi. That is the second, third, etc. , best value that the function can take on and still satisfy the constraints. In Federowicz's algorith, we

just ignore the fact that the two sequences have met and throw out the point where they have crossed and continue on. In Szwarc's method all we need to do is forget that the Boolean constraints were satisfied and continue on listing those that do. It can be seen that not only will all the feasible points of Federowicz's problem (problem I) be listed, but also all the feasible points of problem II as well.

References

[1] Federowicz, Alexander J., "A Generalized Algorithm Solution of a Class of Non-Convex Programming Problems," Ph.D. Thesis in Mathematics, Carnegie Institute of Technology, May, 1963.

[2] Szwarc, Wlodzimierz, "The Mixed Integer Linear Programming Problem When the Integer Variables Are Zero or One," Mathematical Institute of the Polish Academy of Sciences, Wrochaw, Poland and Carnegie Institute of Technology, Graduate School of Industrial Administration, Pittsburgh, Pennsylvania, part of research project on planning and control in Industrial Administration, May 7, 1963.

[3] Tonge, Fred M., "A Revised Algorithm For the Mixed Integer Programming Problem With Boolean Variables," Carnegie Institute of Technology, Graduate School of Industrial Administration, Pittsburgh, Pennsylvania. A research memorandum for private circulation, June 1, 1963.

[4] Garvin, W. W., Introduction to Linear Programming, McGraw-Hill Book Company, Inc., New York, 1960, pp. 220-231.